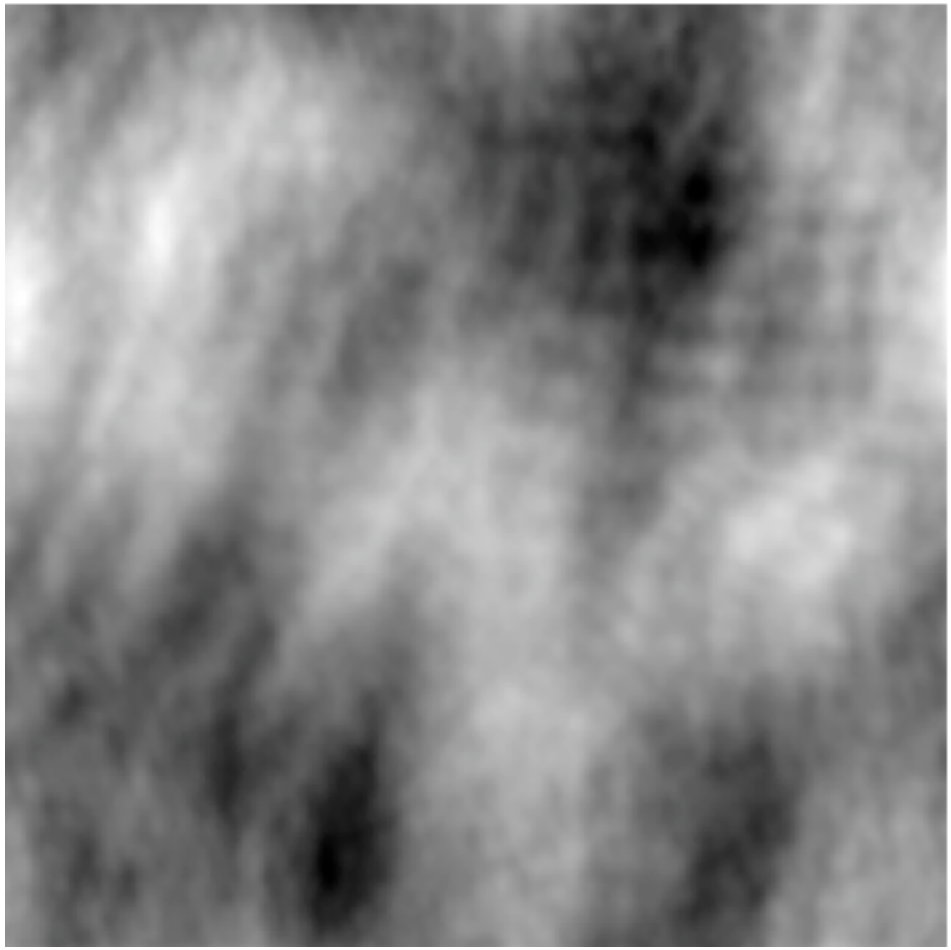
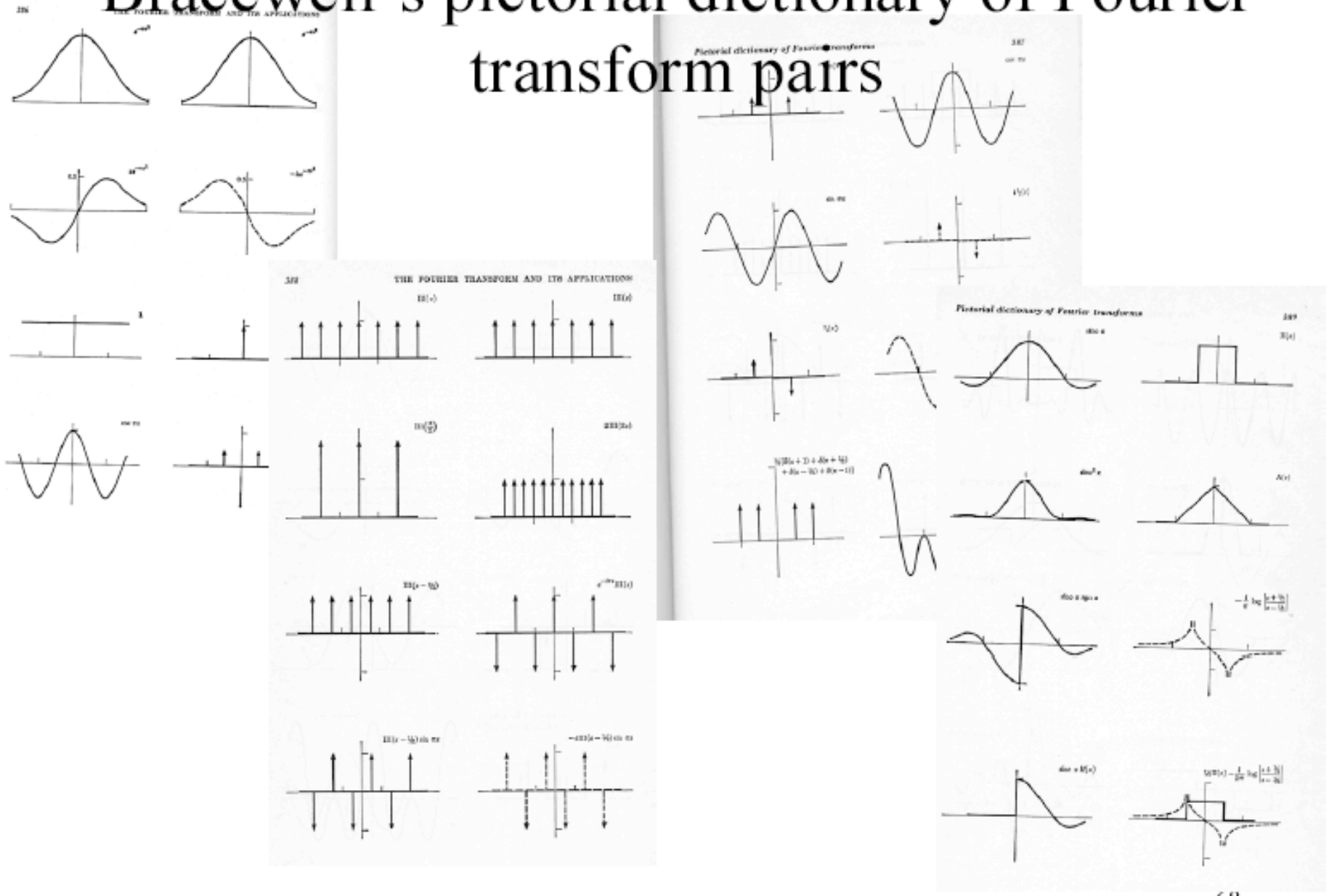
Filters and Edges

Zebra convolved with Leopard



Bracewell's pictorial dictionary of Fourier transform pairs



Bracewell, The Fourier Transform and its Applications, McGraw Hill 1978

Fourier transform of convolution

$$f = g \otimes h$$

$$F[m,n] = DFT(g \otimes h)$$

$$F[m,n] = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k, v-l]h[k,l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

$$= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\sum_{k,l} g[u-k, v-l]e^{-\pi i\left(\frac{um}{M}+\frac{vn}{N}\right)}h[k,l]$$

$$= \sum_{\mu=-k}^{M-k-1}\sum_{\upsilon=-l}^{N-l-1}\sum_{k,l} g[\mu,\upsilon]e^{-\pi i\left(\frac{(k+\mu)m}{M}+\frac{(l+\upsilon)n}{N}\right)}h[k,l]$$

$$= \sum_{k,l} G[m,n]e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}h[k,l]$$

Perform the other DFT (circular boundary conditions)

$$= G[m,n]H[m,n]$$

Analysis of our simple filters

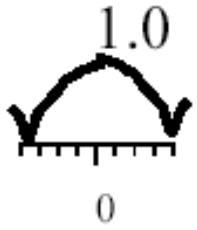
original

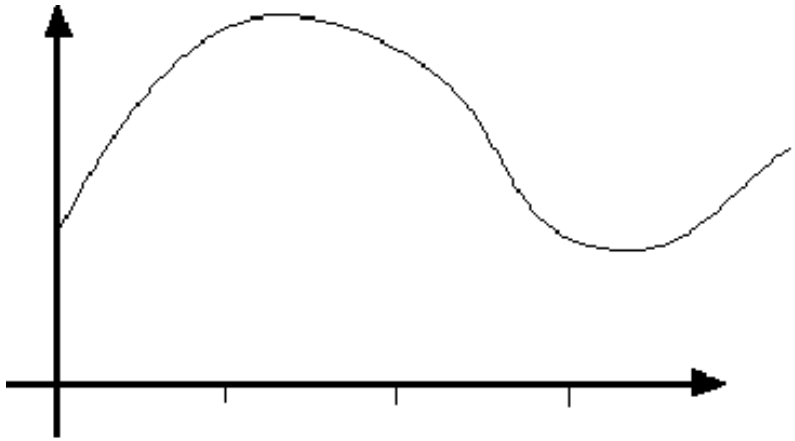
blurred

$$F[m,n] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} f[k,l]e^{-\pi i\left(\frac{km}{M}+\frac{\ln}{N}\right)}$$

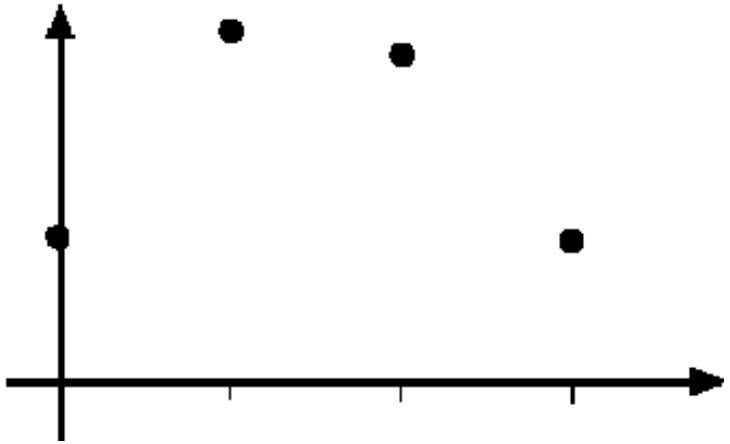
$$= \frac{1}{3}\left(1+2\cos\left(\frac{\pi m}{M}\right)\right)$$

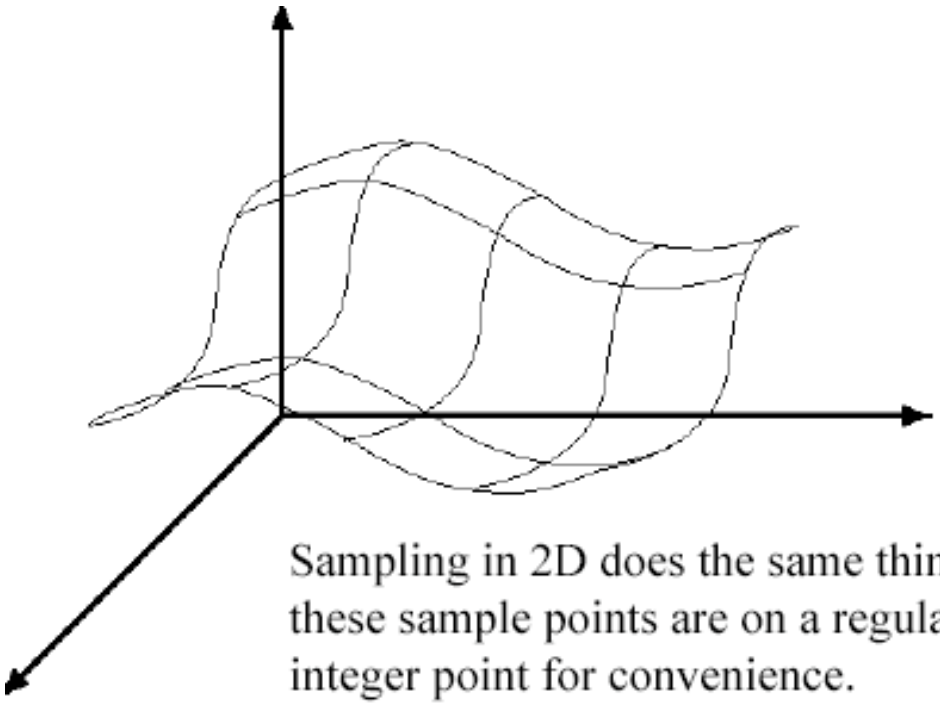
Low-pass filter



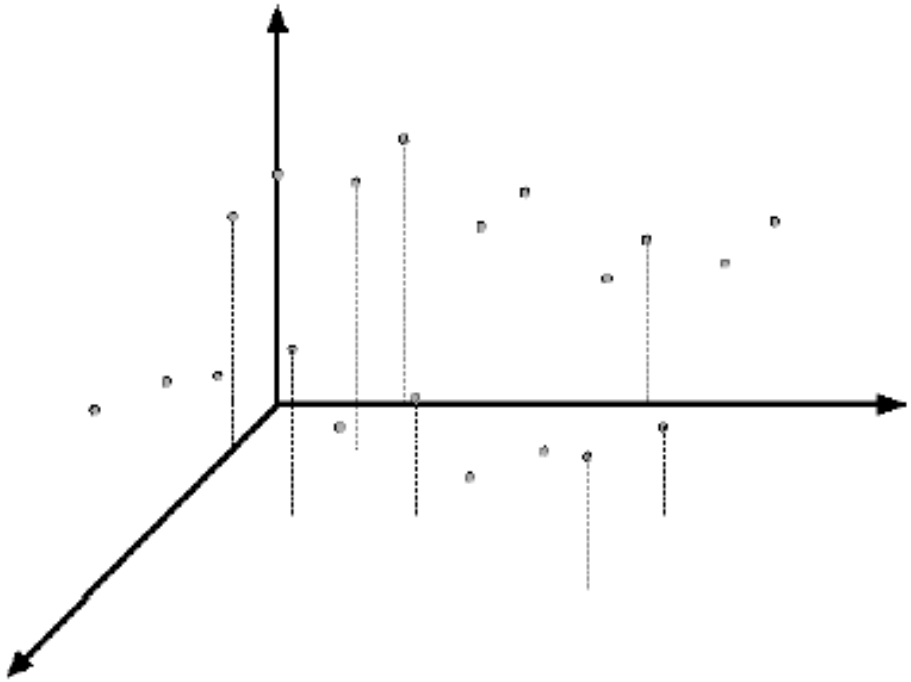


Sampling in 1D takes a continuous function and replaces it with a vector of values, consisting of the function's values at a set of sample points. We'll assume that these sample points are on a regular grid, and can place one at each integer for convenience.





Sampling in 2D does the same thing, only in 2D. We'll assume that these sample points are on a regular grid, and can place one at each integer point for convenience.



A continuous model for a sampled function

- We want to be able to approximate integrals sensibly
- Leads to
  - the delta function
  - model on right

$$\text{Sample}_{2D}(f(x,y)) = \sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty} f(x,y)\delta(x-i,y-j)$$

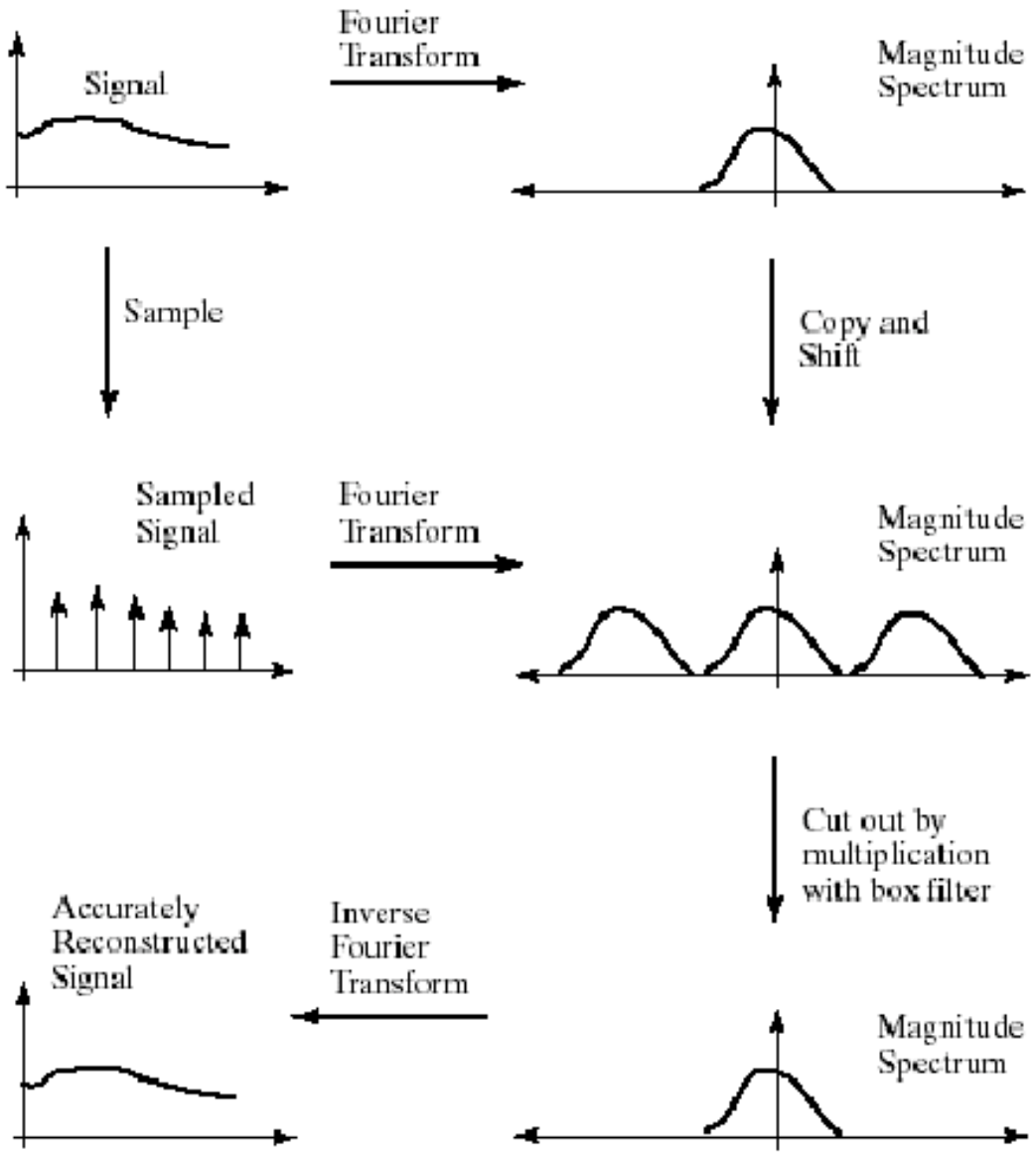
$$= f(x,y)\sum_{i=-\infty}^{\infty}\sum_{i=-\infty}^{\infty}\delta(x-i,y-j)$$

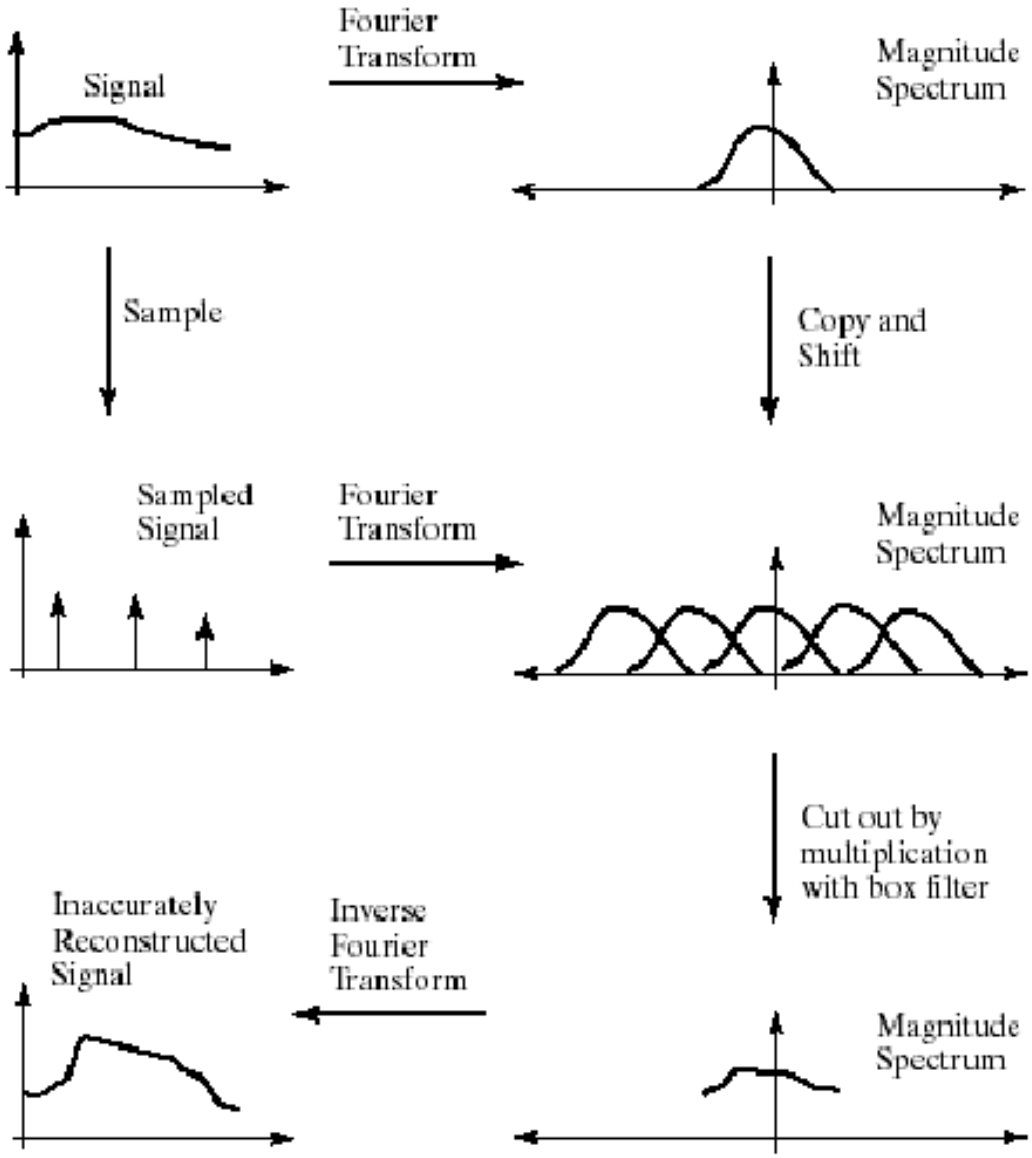
The Fourier transform of a sampled signal

$$F(\text{Sample}_{2D}(f(x,y))) = F\left(f(x,y) \sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j) \right)$$

$$= F(f(x,y)) ** F\left(\sum_{i=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \delta(x-i, y-j) \right)$$

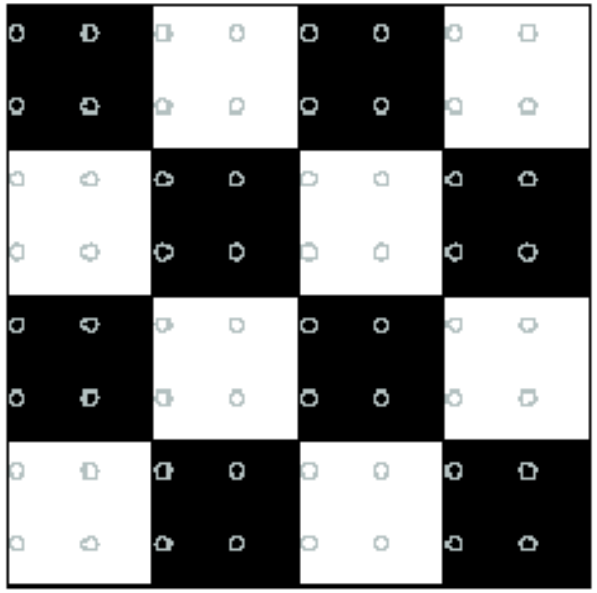
$$= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} F(u-i, v-j)$$

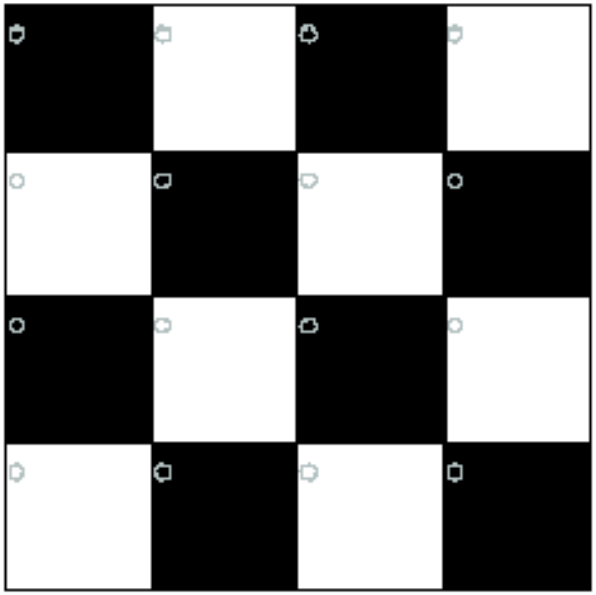

Signal
Fourier Transform
Magnitude Spectrum
Sample
Copy and Shift
Sampled Signal
Fourier Transform
Magnitude Spectrum
Cut out by multiplication with box filter
Accurately Reconstructed Signal
Inverse Fourier Transform
Magnitude Spectrum

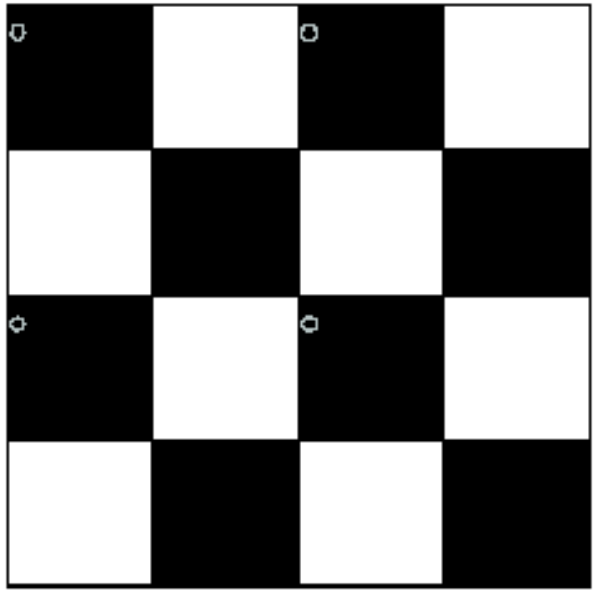

Signal
Fourier
Transform
Magnitude
Spectrum
Sample
Copy and
Shift
Sampled
Signal
Fourier
Transform
Magnitude
Spectrum
Cut out by
multiplication
with box filter
Inaccurately
Reconstructed
Signal
Inverse
Fourier
Transform
Magnitude
Spectrum

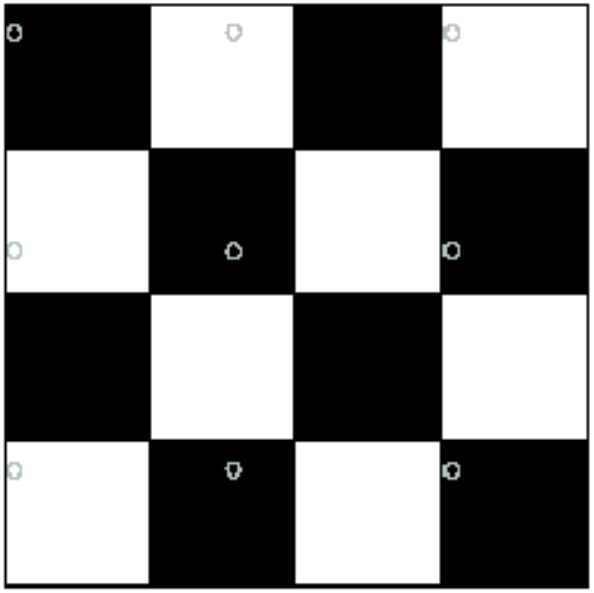
Aliasing

- Can't shrink an image by taking every second pixel
- If we do, characteristic errors appear
  - In the next few slides
  - Typically, small phenomena look bigger; fast phenomena can look slower
  - Common phenomenon
    - Wagon wheels rolling the wrong way in movies
    - Checkerboards misrepresented in ray tracing
    - Striped shirts look funny on colour television

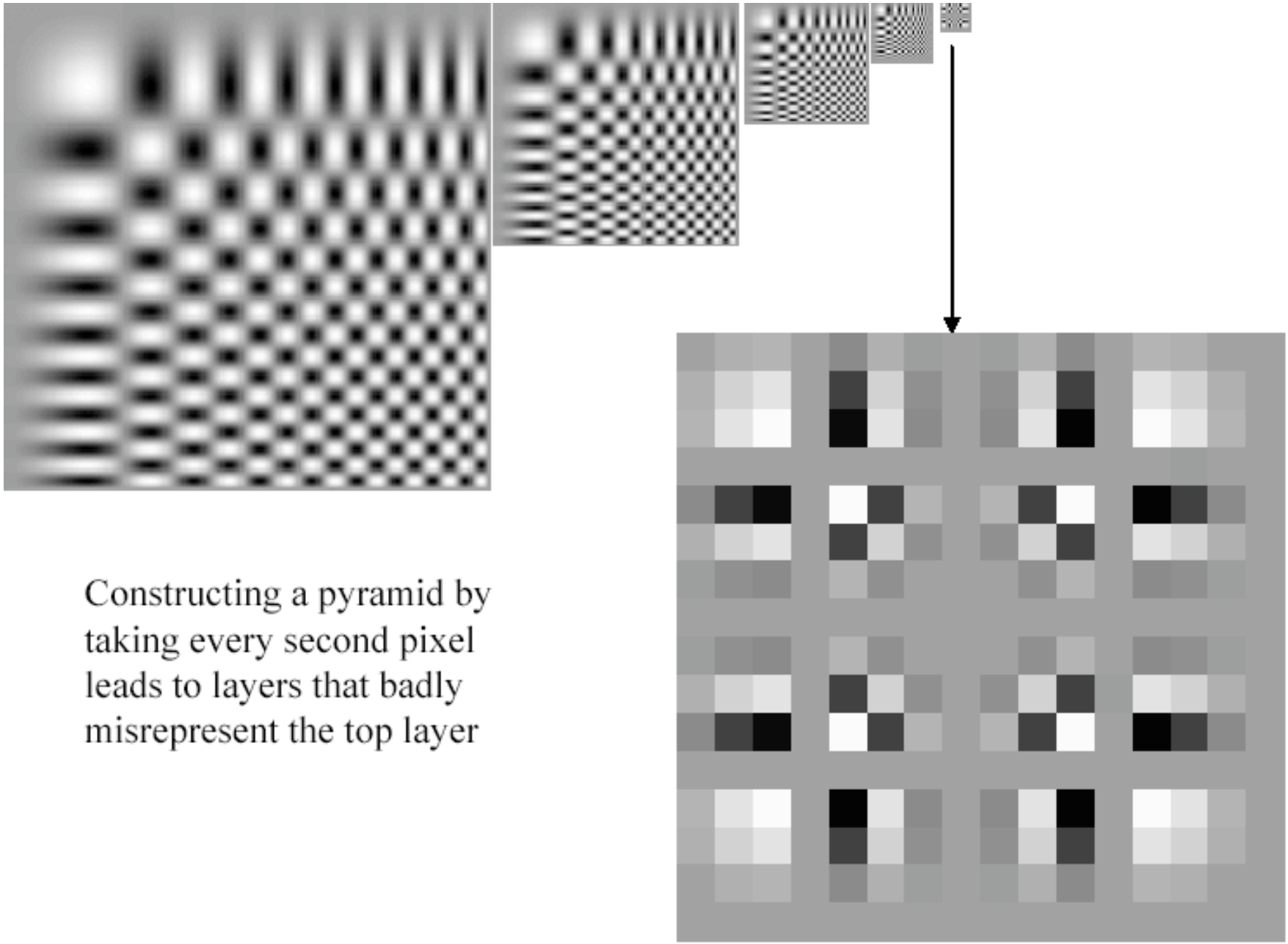






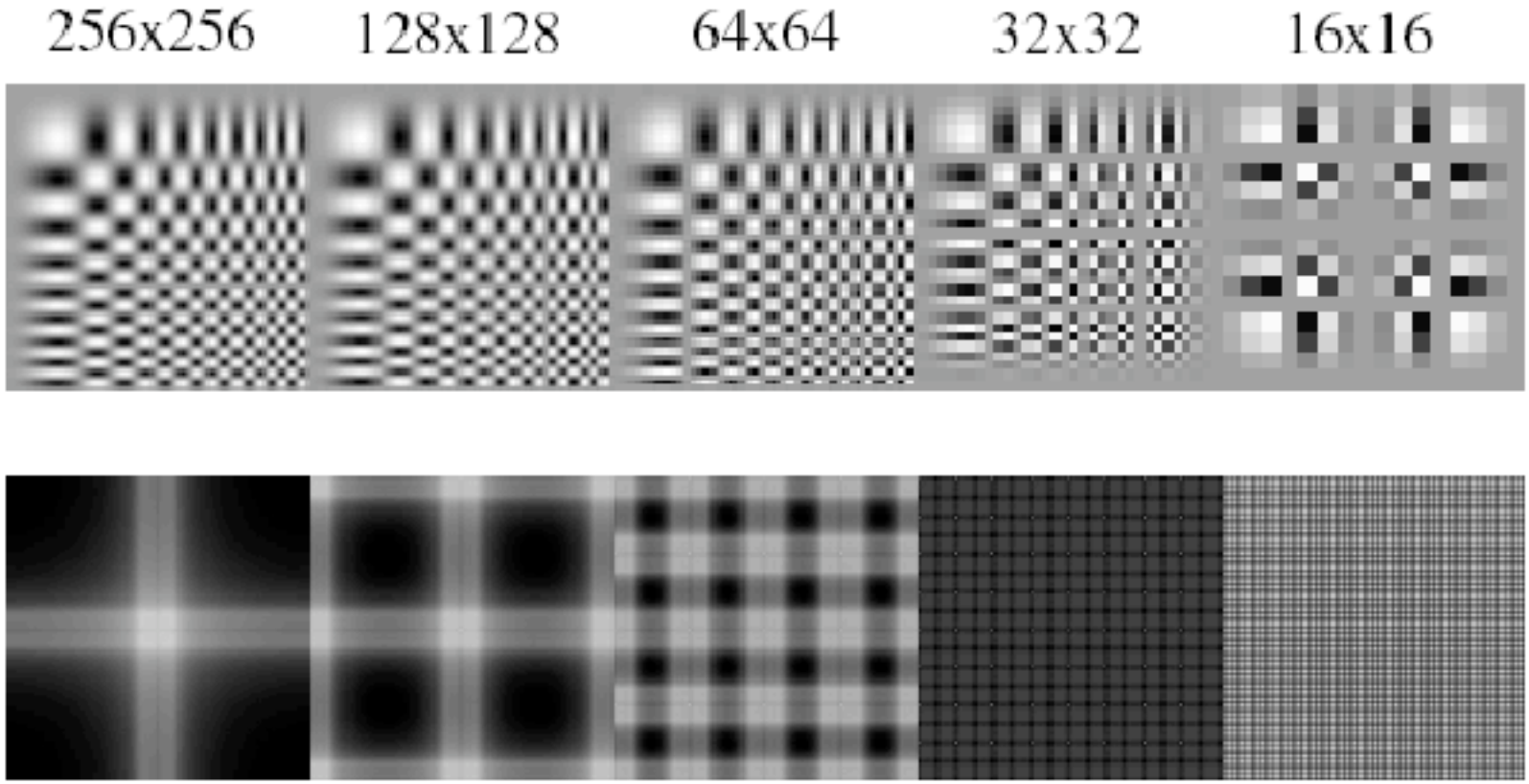


Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable.
Top right also yields a reasonable representation.
Bottom left is all black (dubious) and bottom right has checks that are too big.



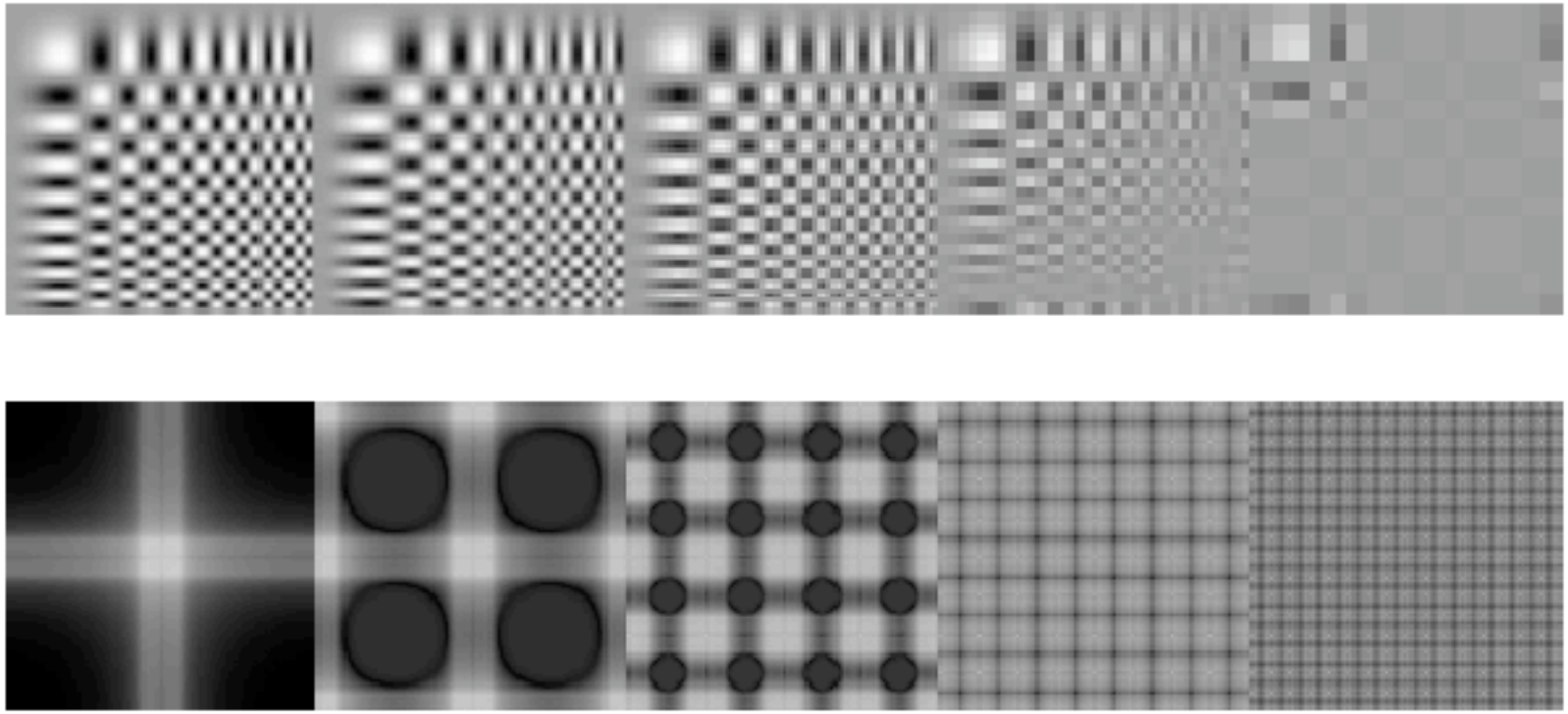
Constructing a pyramid by taking every second pixel leads to layers that badly misrepresent the top layer

Sampling without smoothing. Top row shows the images, sampled at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.



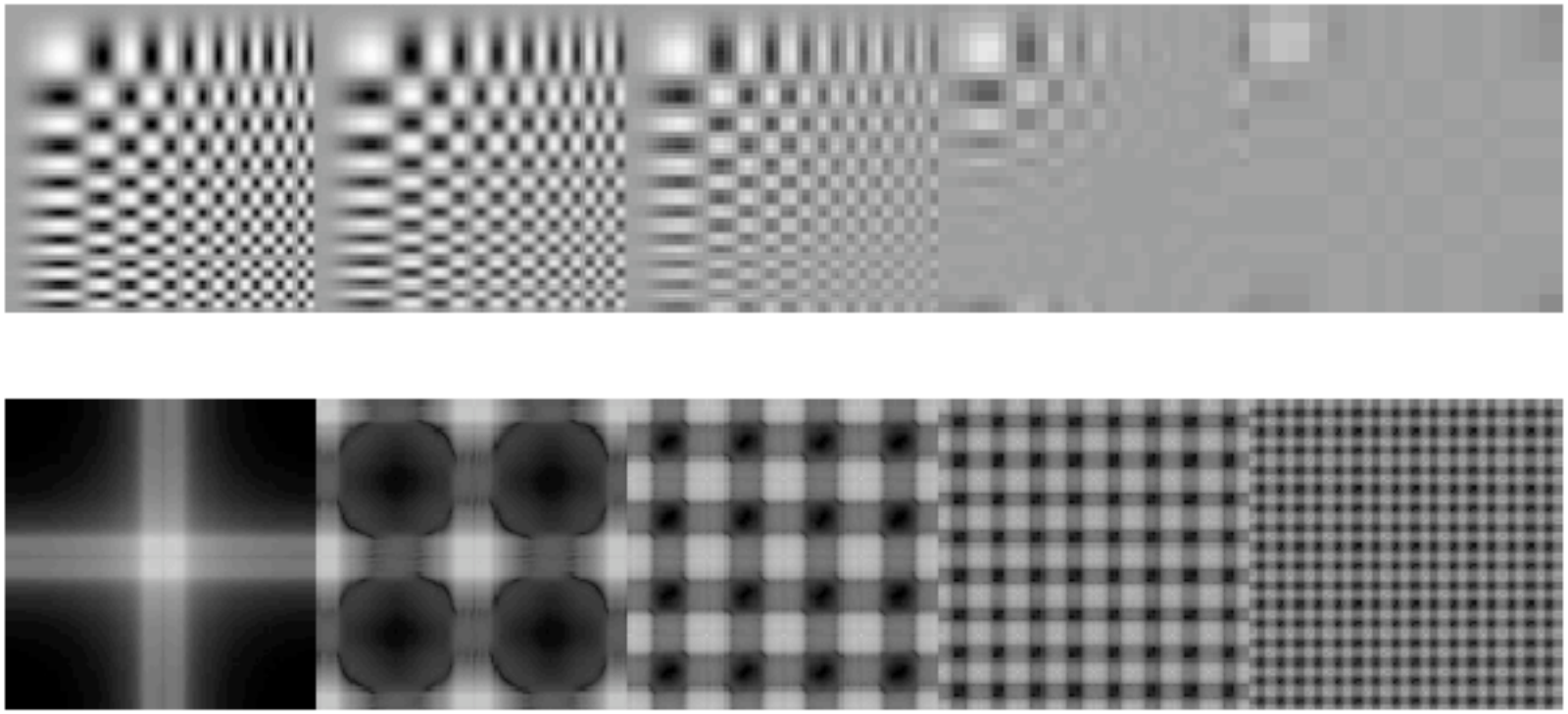
Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1 pixel, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.

256x256 128x128 64x64 32x32 16x16

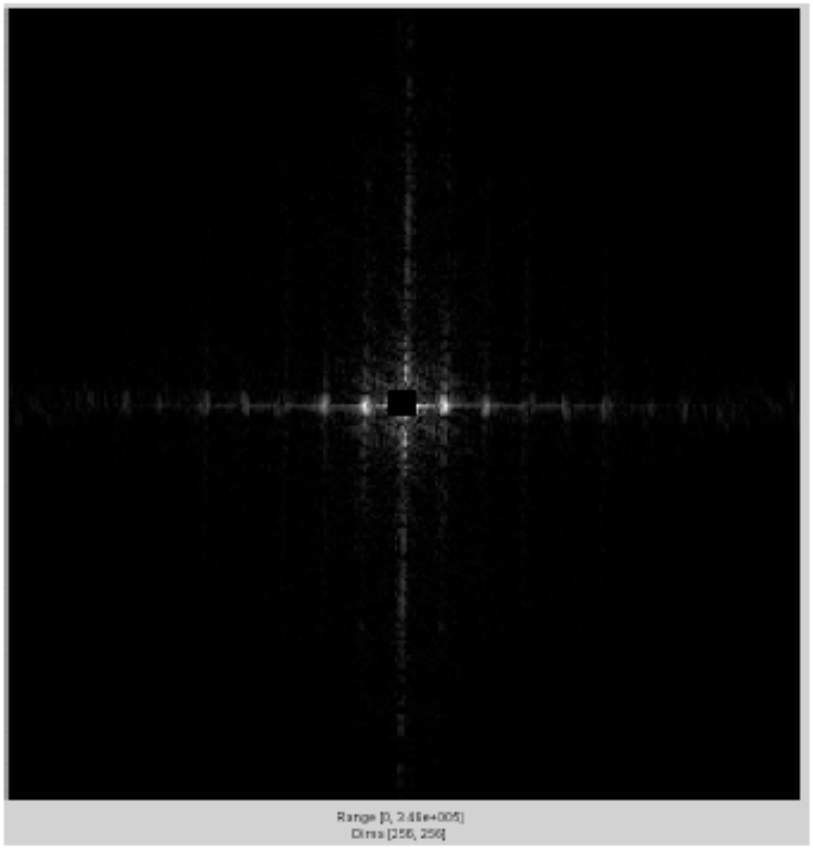


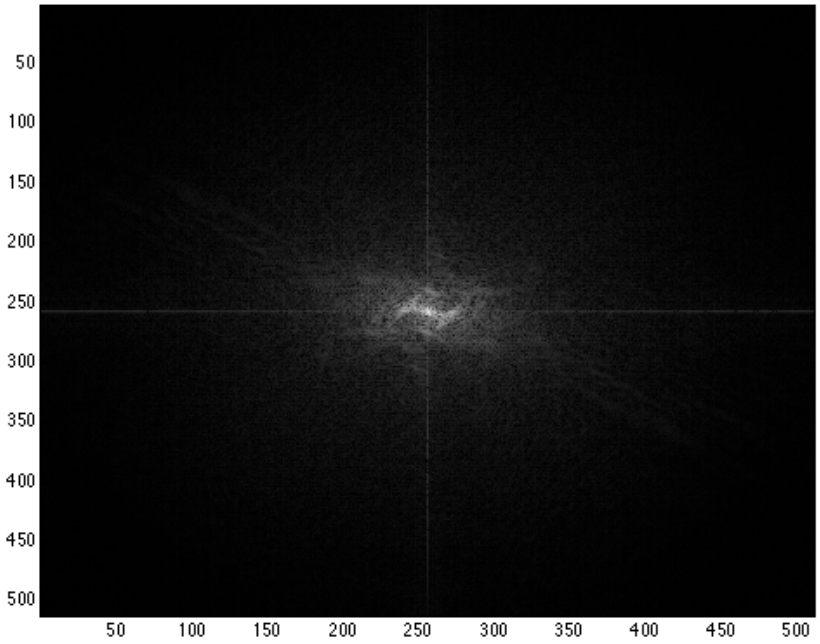
Sampling with smoothing. Top row shows the images. We get the next image by smoothing the image with a Gaussian with sigma 1.4 pixels, then sampling at every second pixel to get the next; bottom row shows the magnitude spectrum of these images.

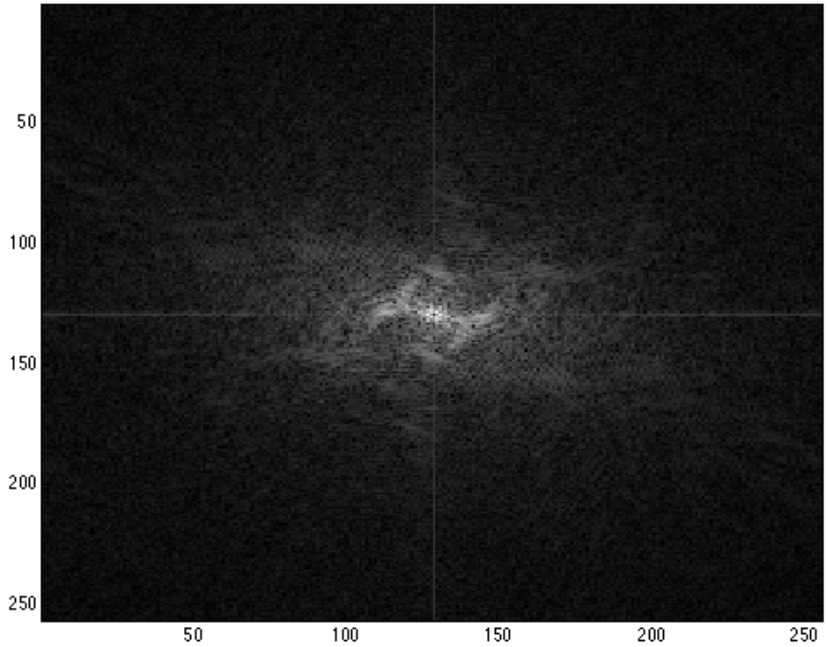
256x256 128x128 64x64 32x32 16x16

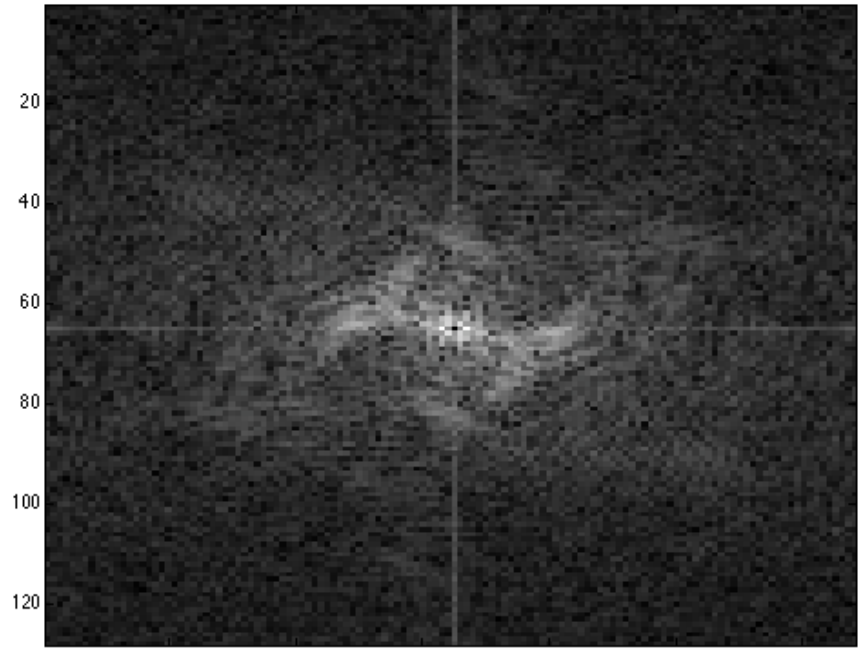


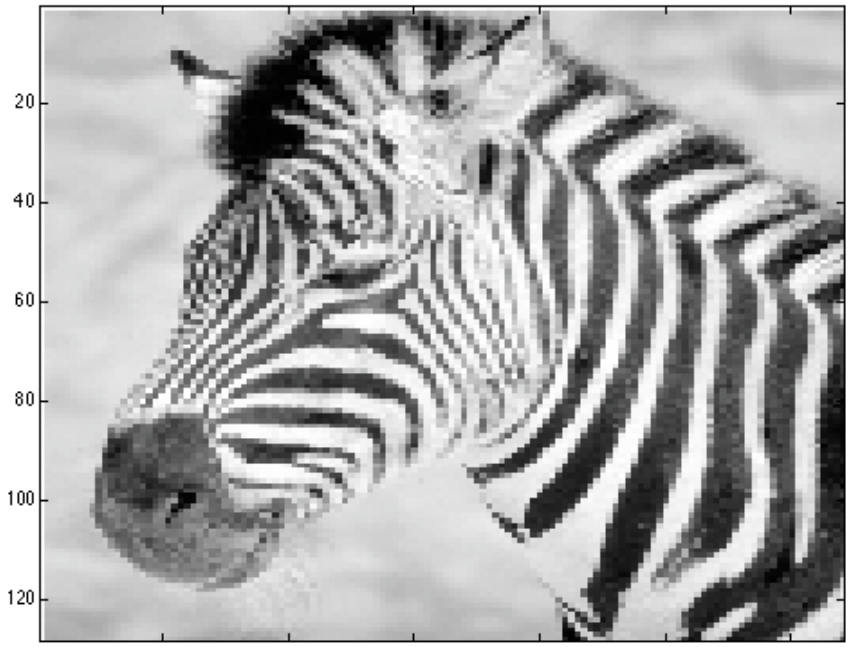
Fourier transform magnitude

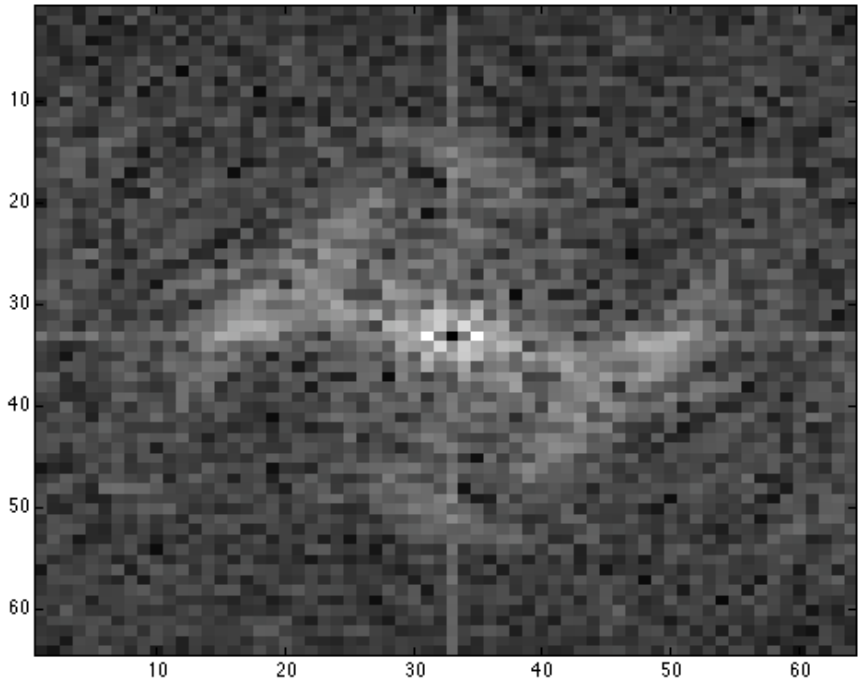


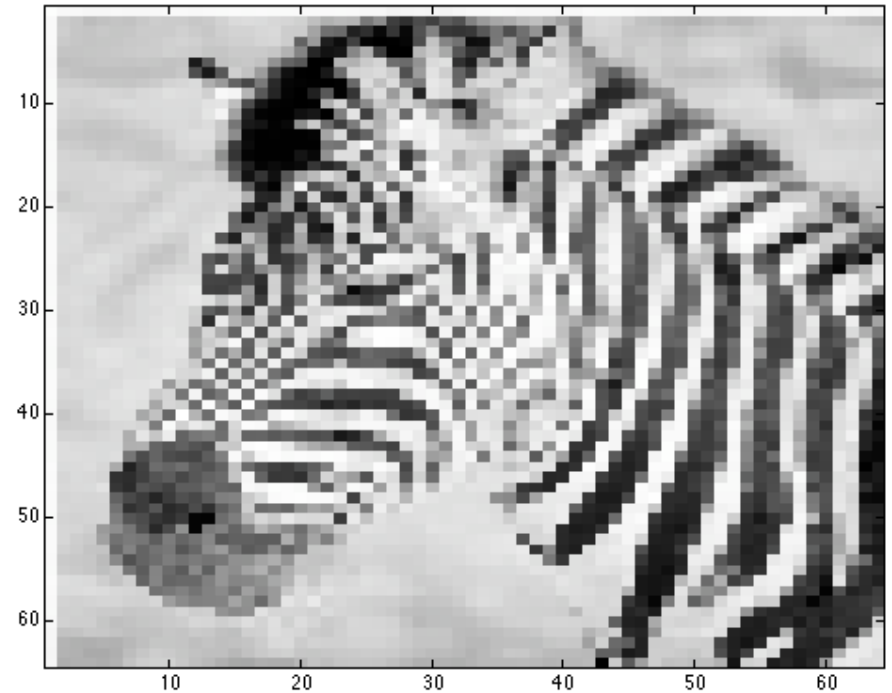

50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

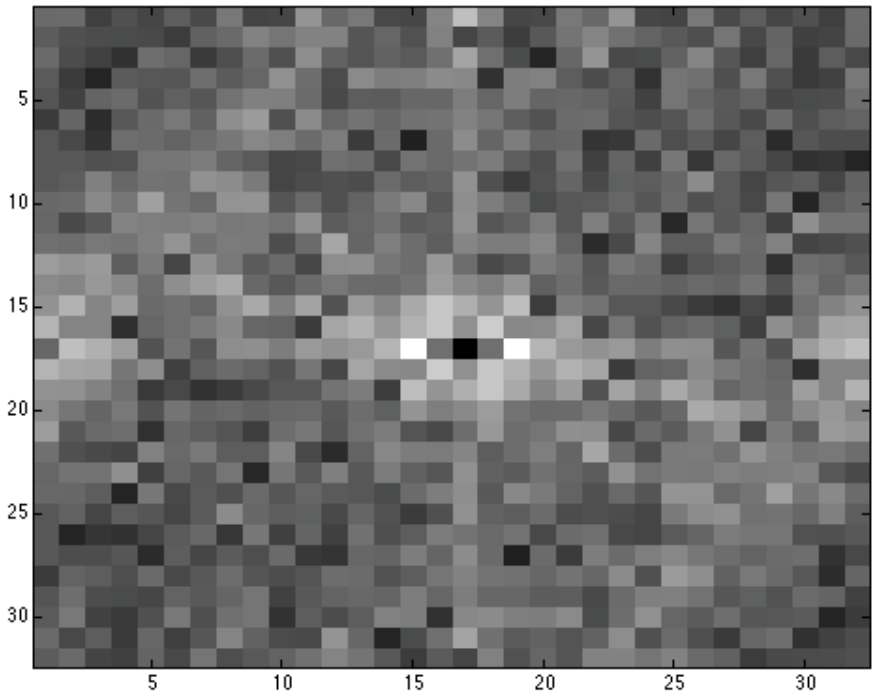

50
100
150
200
250
50
100
150
200
250

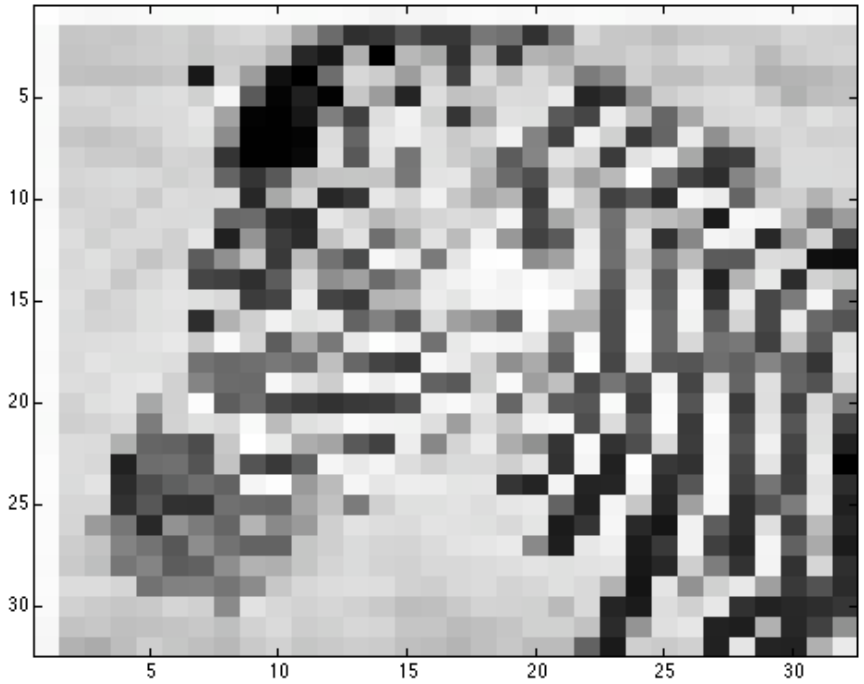

20
40
60
80
100
120
20
40
60
80
100
120

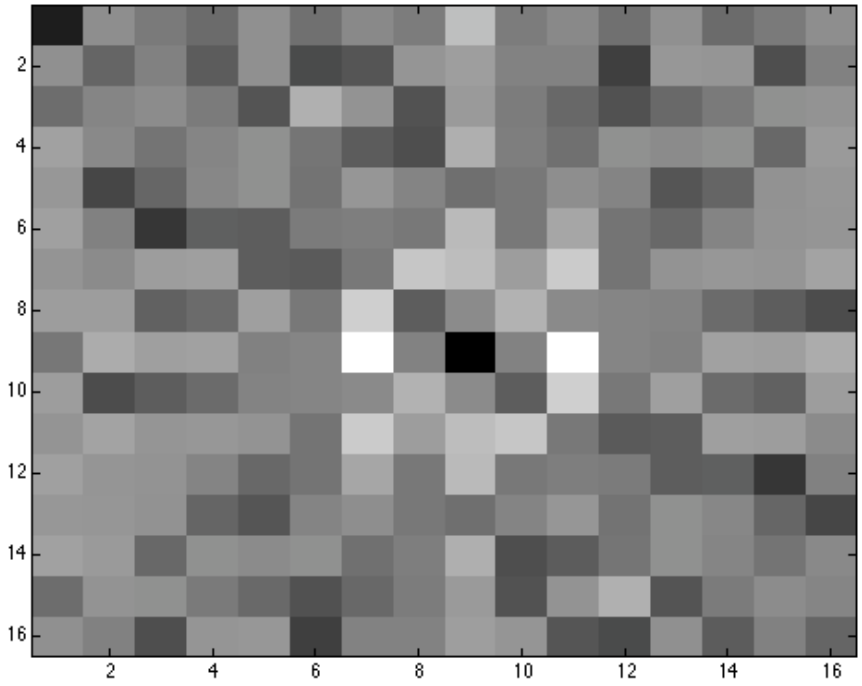

20
40
60
80
100
120
20
40
60
80
100
120

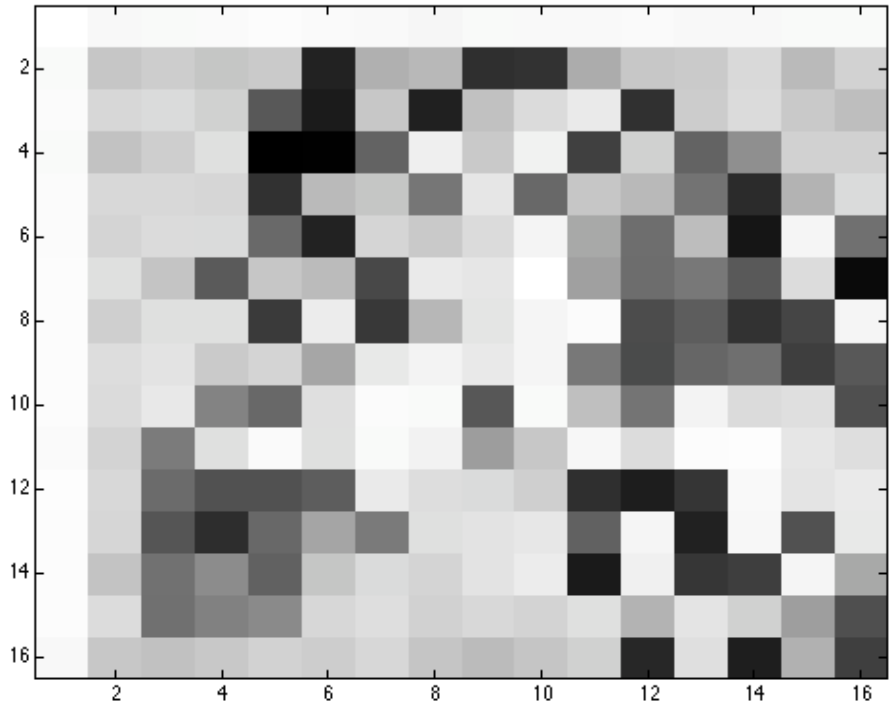

10
20
30
40
50
60
10
20
30
40
50
60


10
20
30
40
50
60
10
20
30
40
50
60

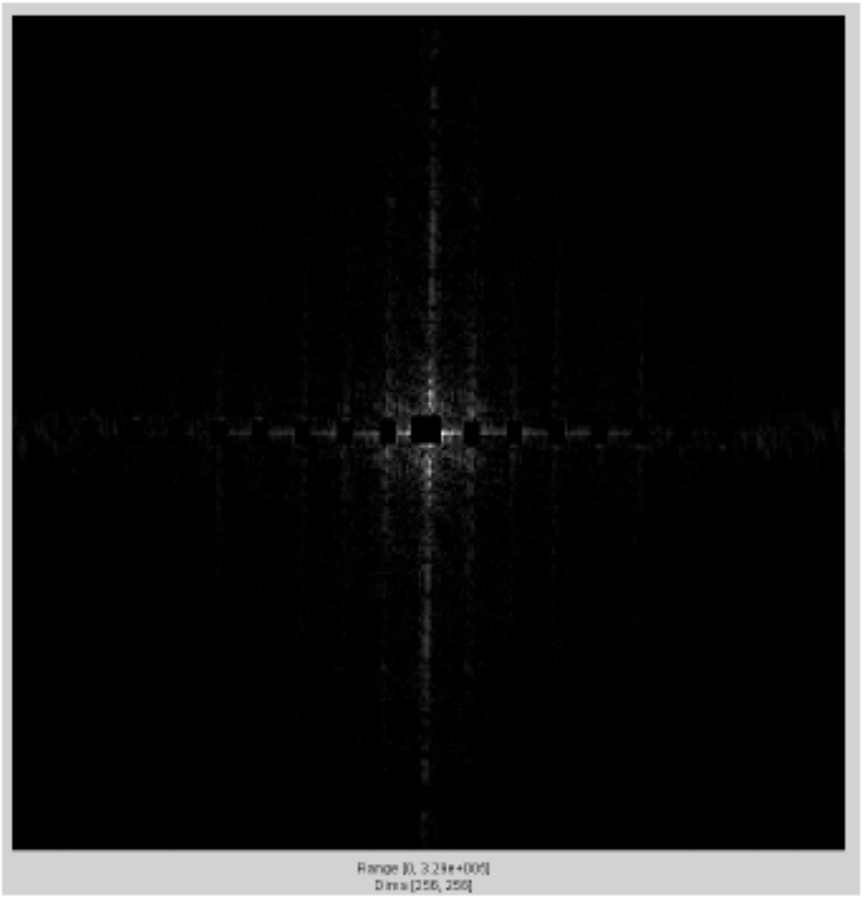

5
10
15
20
25
30
5
10
15
20
25
30

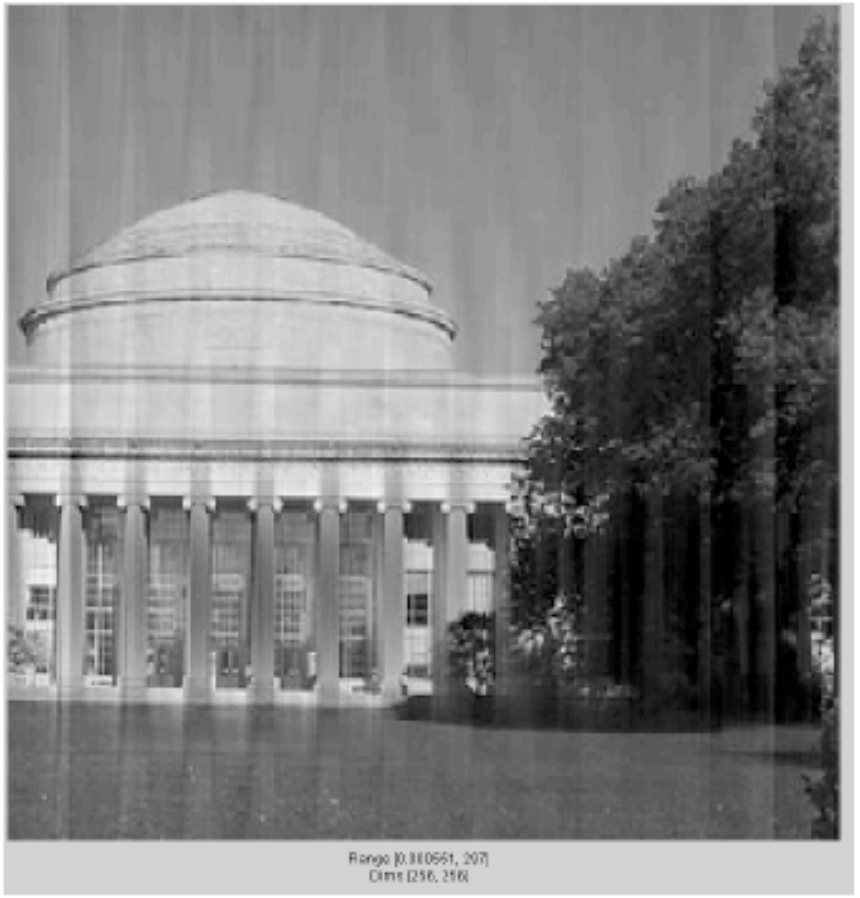

5
10
15
20
25
30
5
10
15
20
25
30


2
4
6
8
10
12
14
16
2
4
6
8
10
12
14
16


2
4
6
8
10
12
14
16
2
4
6
8
10
12
14
16

Masking out the fundamental and harmonics from periodic pillars





Edge Detection as a signal detection problem

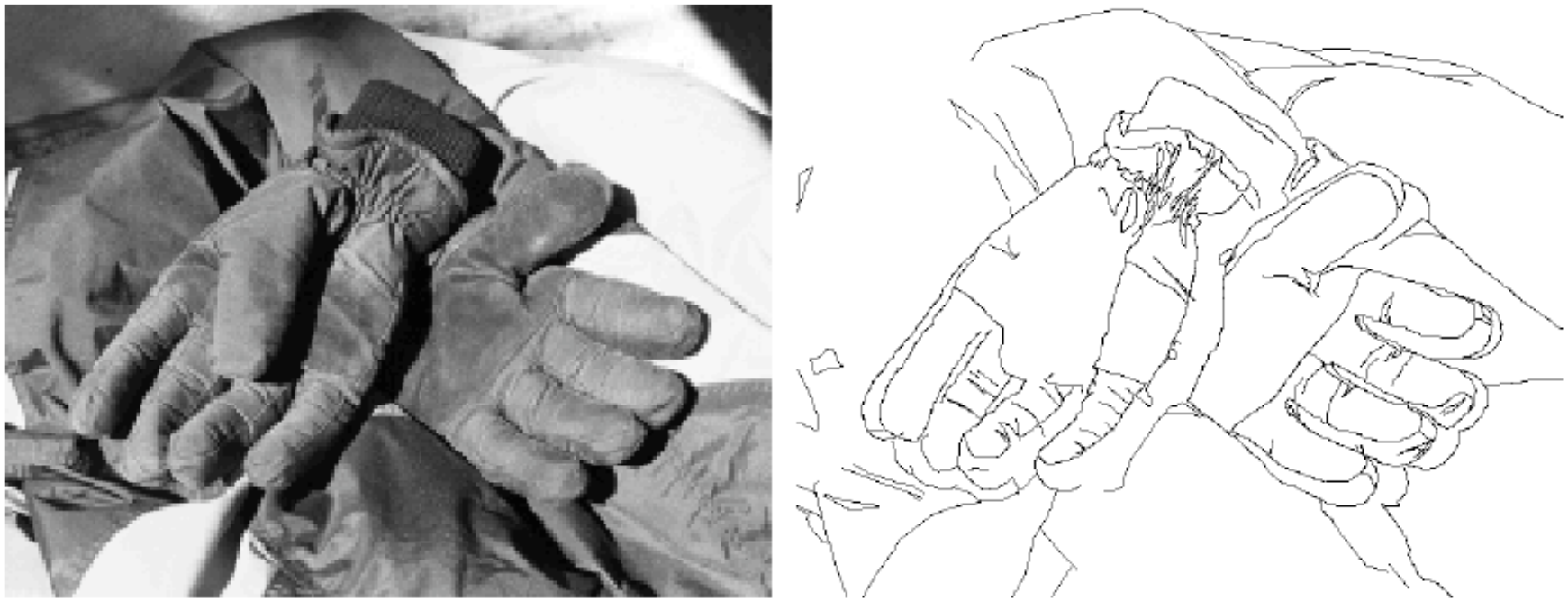


Figure 2.4 A typical image (left) and the ground truth segmentation (right). courtesy of K. Bowyer at U. South Florida.

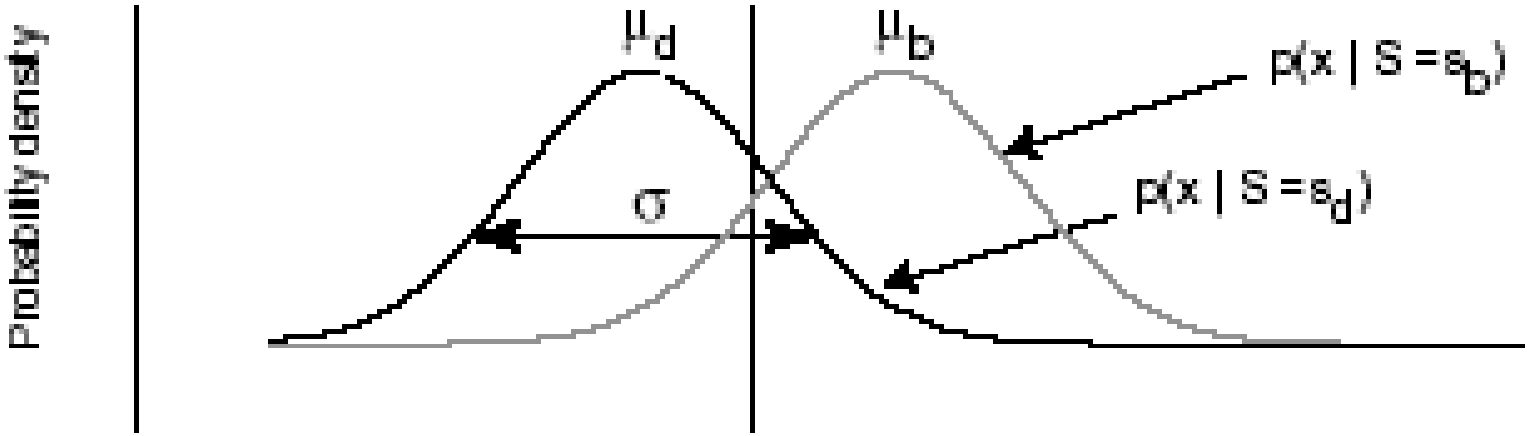
Goal: find meaningful intensity boundaries.

$$\log \frac{P(\phi(I(x))|\ \text{on-edge})}{P(\phi(I(x))|\ \text{off-edge})} > T,$$

Simplest Model: (Canny)

Edge(x) = $a\ U(x) + n(x)$

Convolve image with U and find points with high magnitude. Choose value by comparing with a threshold determined by the noise model.



Probability of a filter response off an edge

Probability of a filter response on an edge

Fundamental limits on edge detection

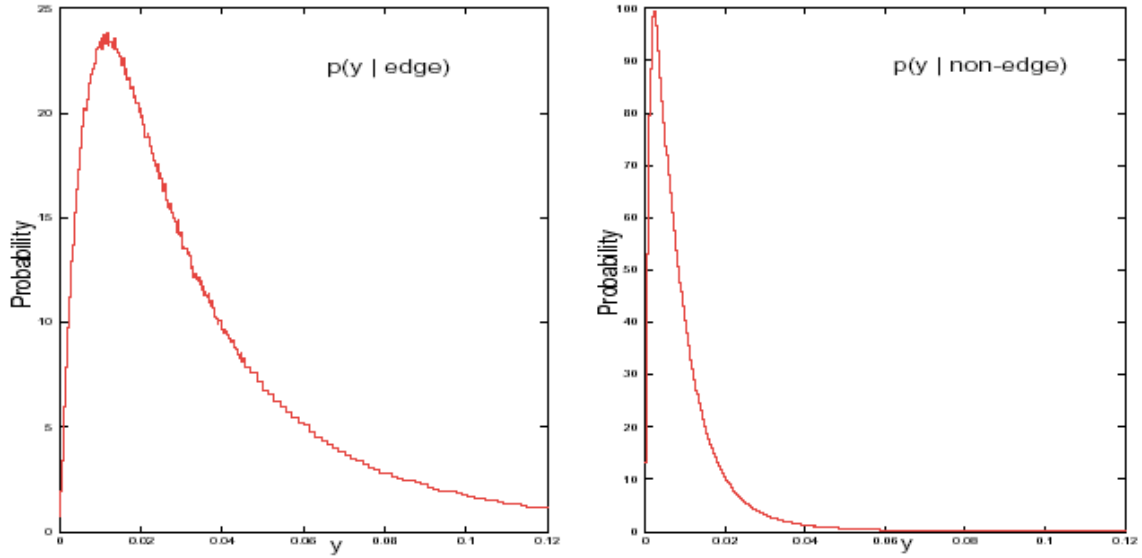


Figure 2.5 Empirical distributions for a gradient filter response on boundaries (left) and off boundaries (right).

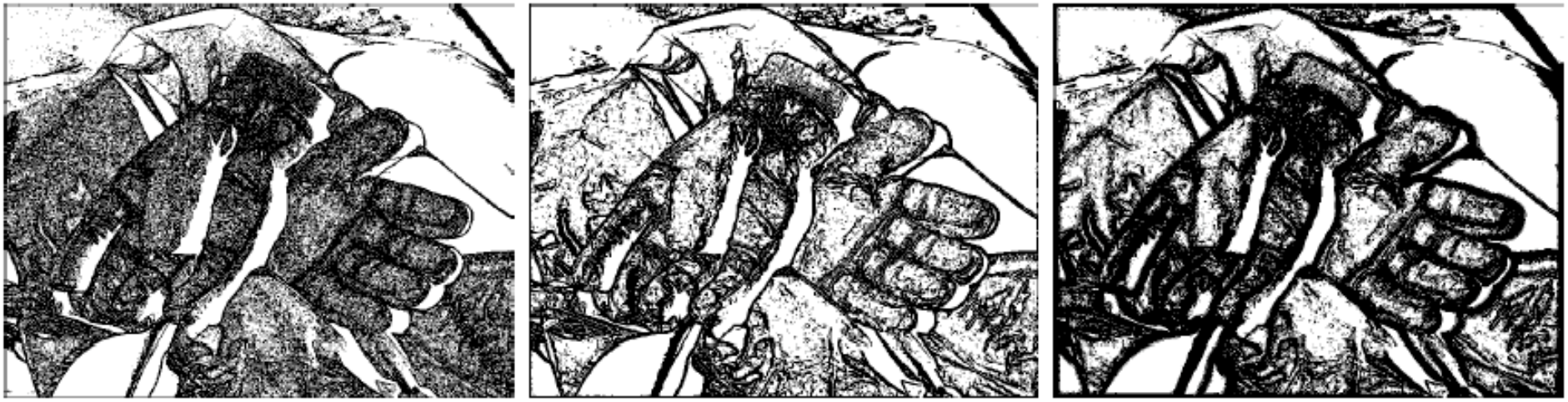
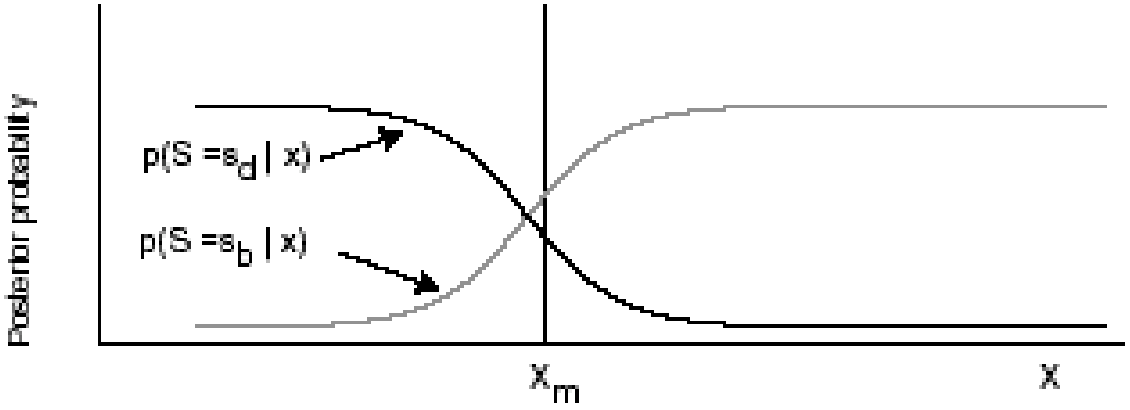


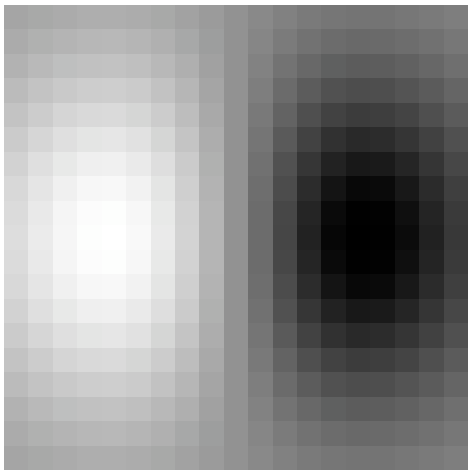
Figure 2.6 The edge estimated on the glove image by ML with the filter at scale 0 (left), filter at scale 1 (centre), and filter with scales $0, 1, 2, 4$ (right). Observe that ML significantly overestimates the number of edges in this image.

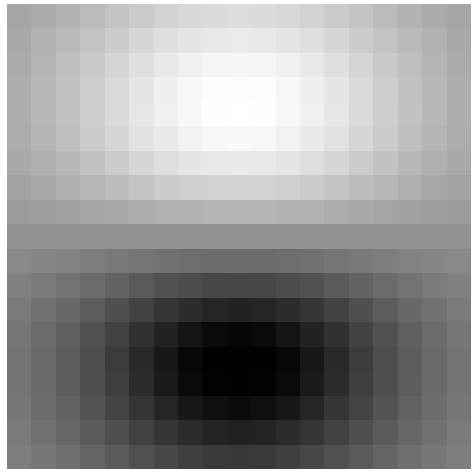
Need to take into account base edge rate.

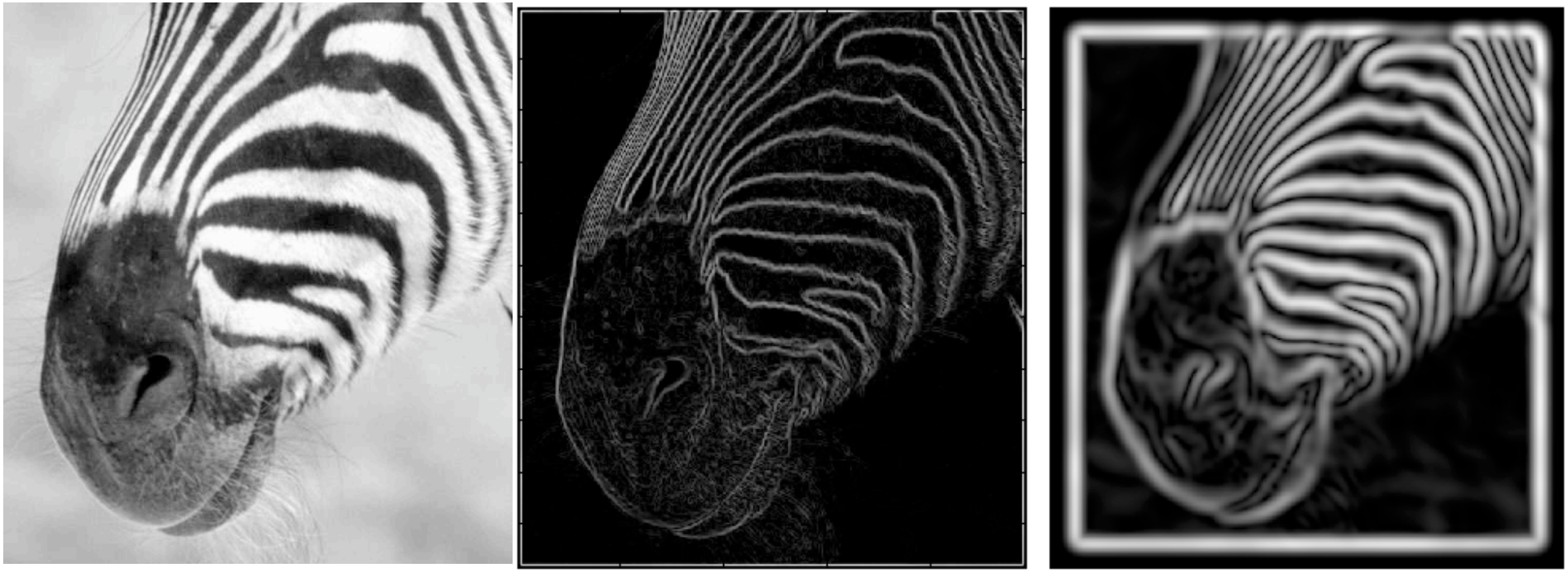


Smoothing and Differentiation

- Issue: noise
  - smooth before differentiation
  - two convolutions to smooth, then differentiate?
  - actually, no - we can use a derivative of Gaussian filter
    - because differentiation is convolution, and convolution is associative

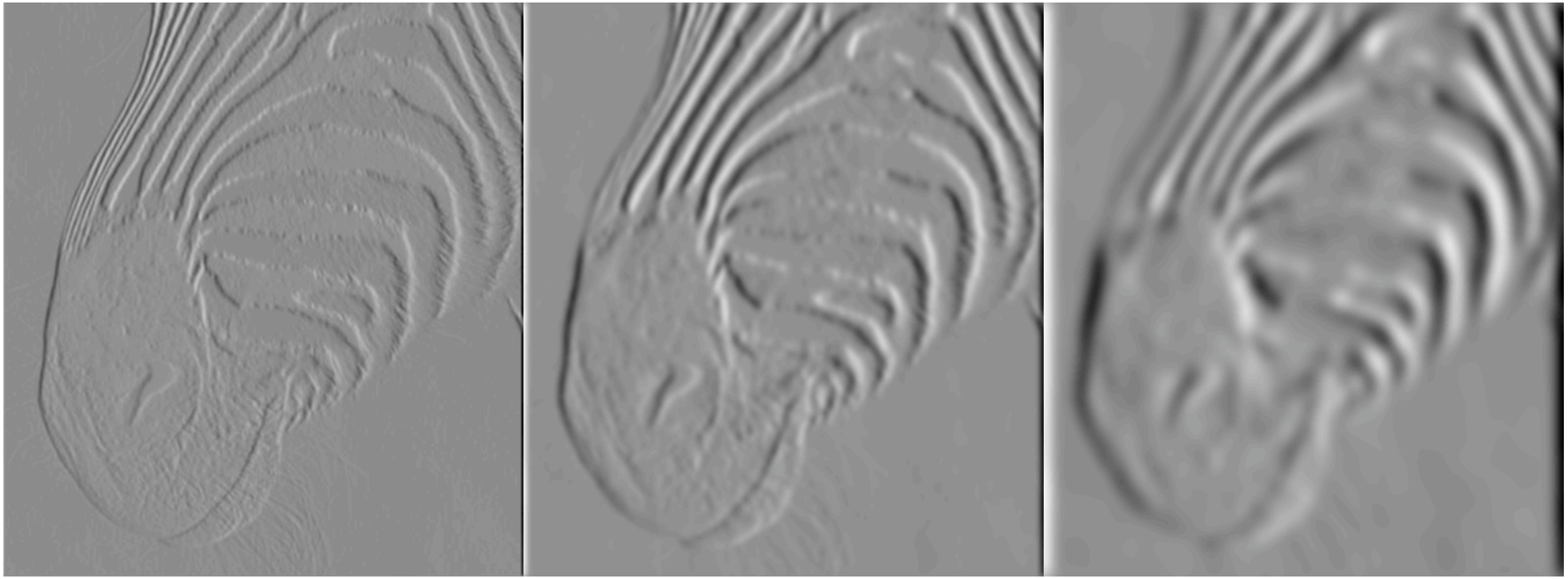






There are three major issues:

1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trail; how do we identify the significant points?
3) How do we link the relevant points up into curves?



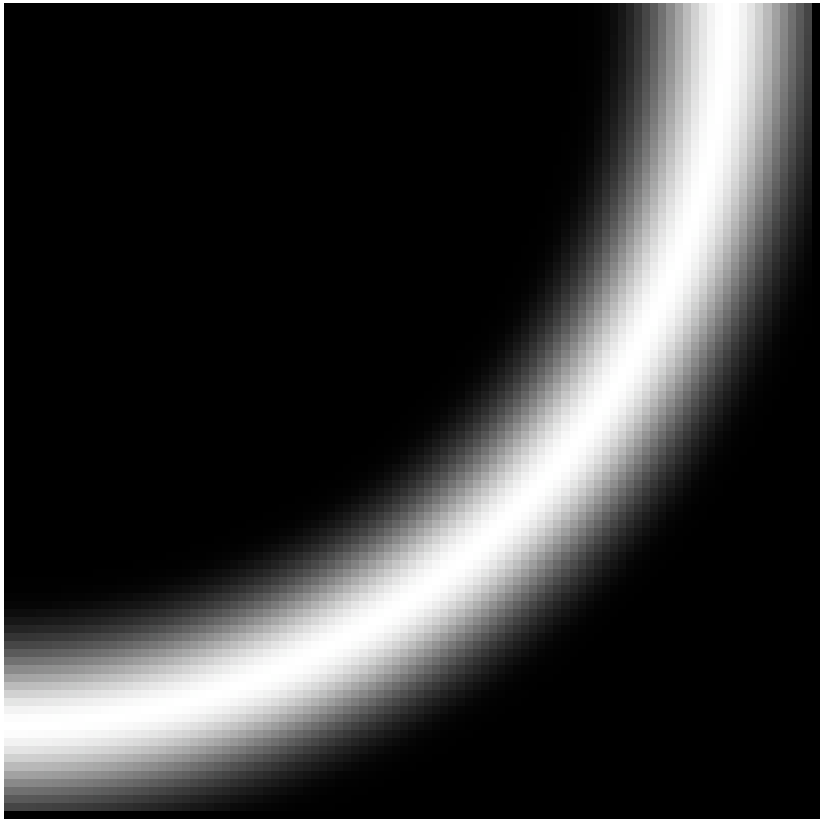
1 pixel

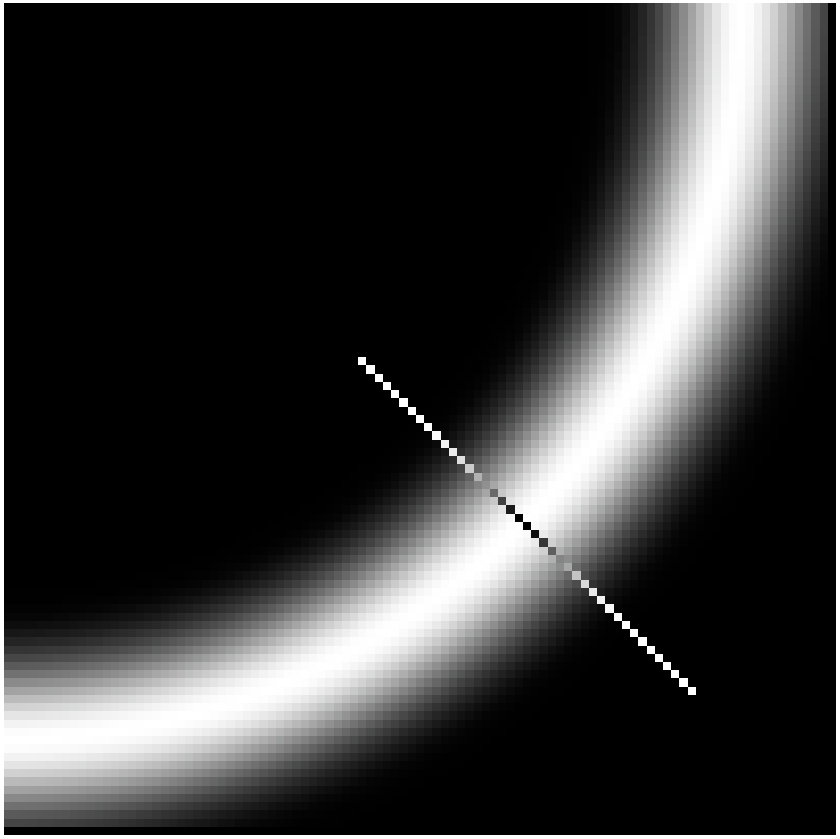
3 pixels

7 pixels

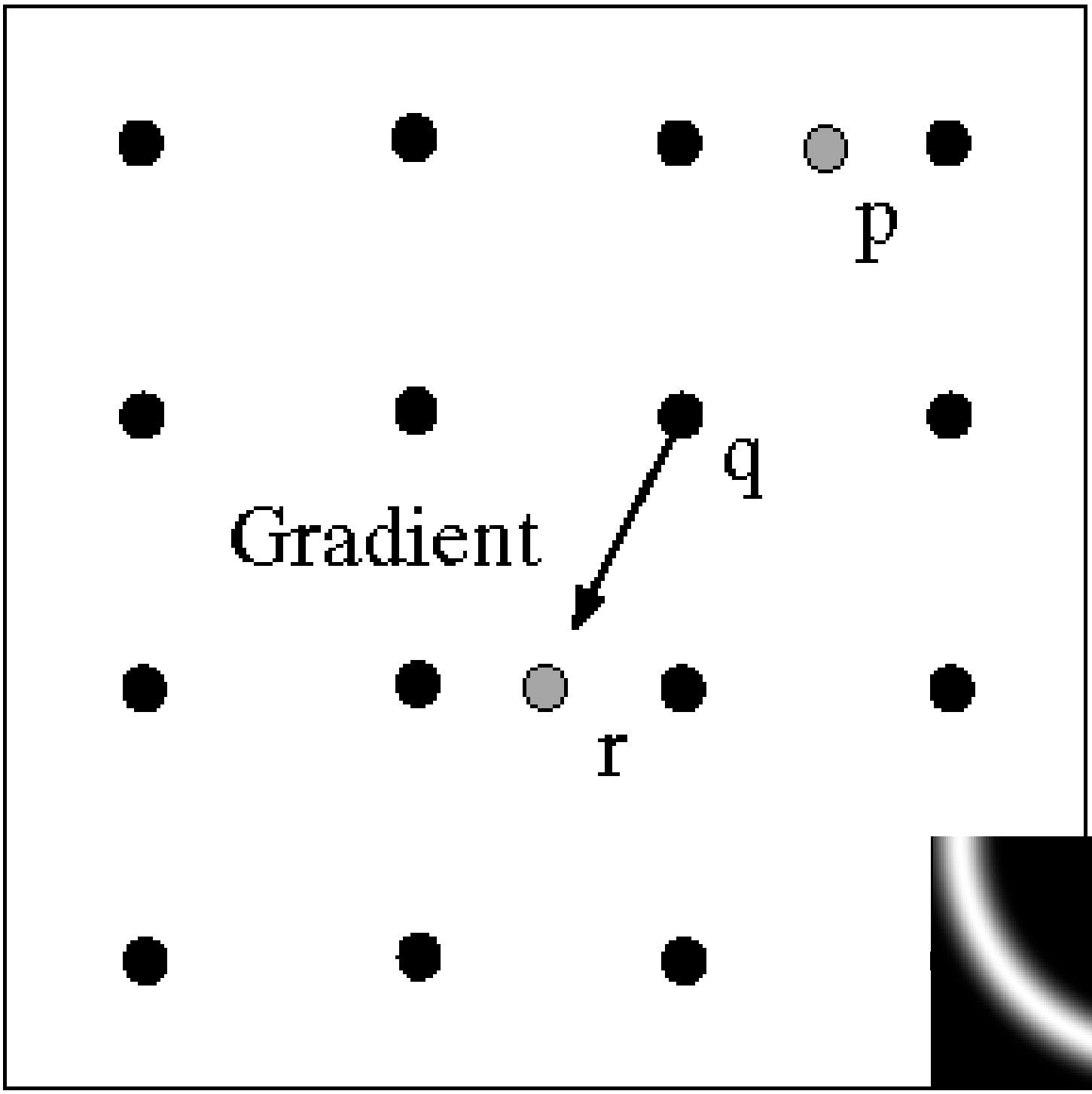
The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.

Computing Edges via the gradient



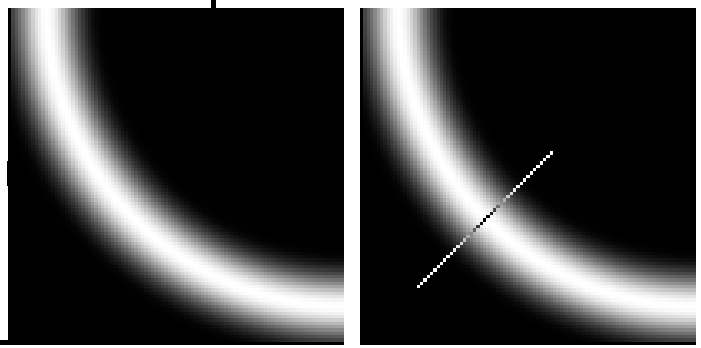


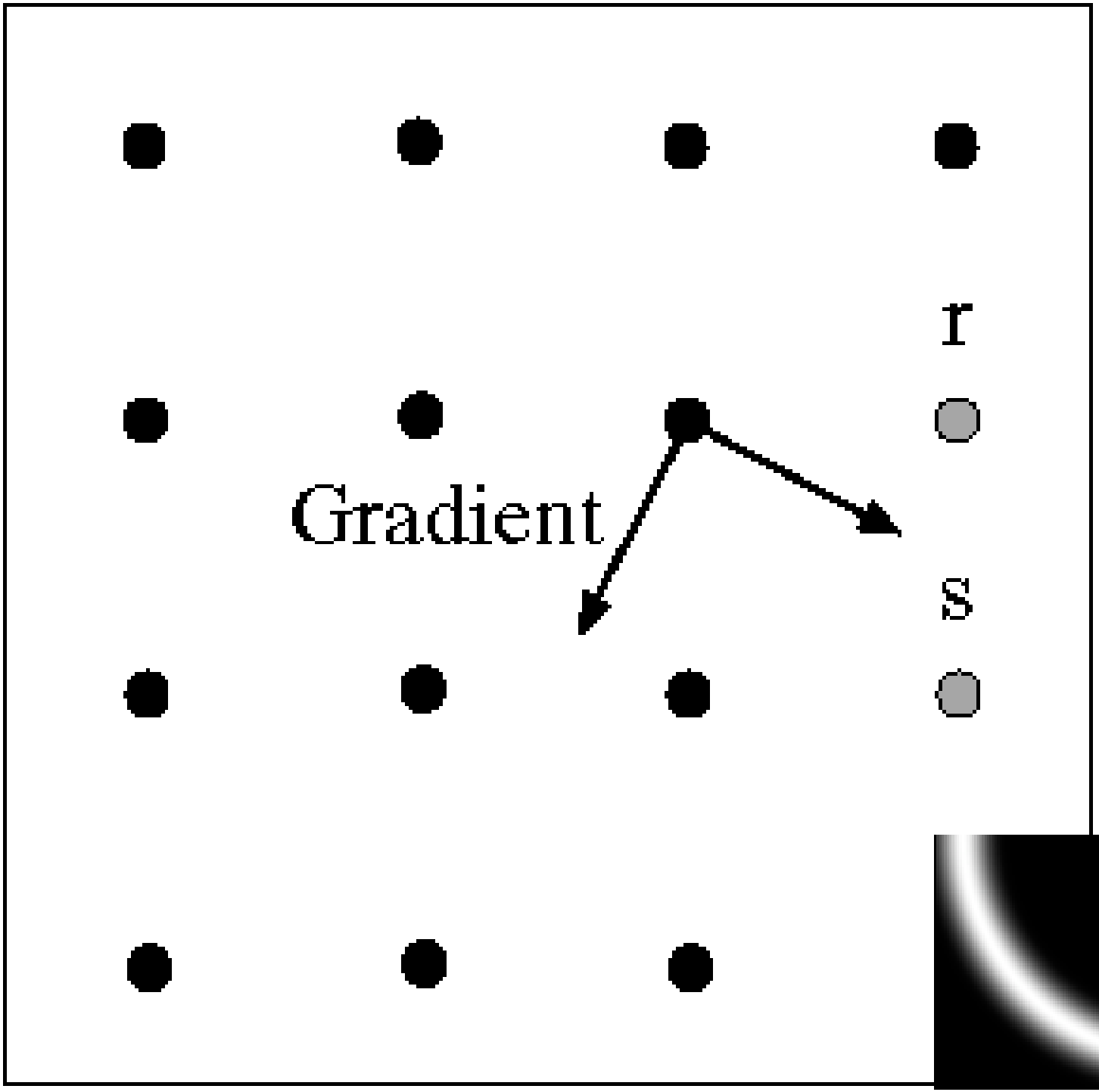
We wish to mark points along the curve where the magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?



Non-maximum suppression

At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.





Predicting the next edge point

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).